Australian Government
Department of Defence
Defence Science and
Technology Organisation

Multi-camera Human Tracking via Clustering

Dmitri Kamenetsky

Intelligence, Surveillance and Reconnaissance Division

Defence Science and Technology Organisation

DSTO–TR–2663

ABSTRACT

We describe an efficient method for tracking humans in a multi-camera network. Based on online clustering, the proposed method groups single-camera tracks into mutually exclusive subsets corresponding to individuals. Experiments indicate that the algorithm is capable of achieving strong tracking accuracy on simulated data.

Published by

DSTO Defence Science and Technology Organisation
PO Box 1500
Edinburgh, South Australia 5111, Australia

Telephone: (08) 7389 5555
Facsimile: (08) 7389 6567

AR No. AR 015-218
November 2011

APPROVED FOR PUBLIC RELEASE

Multi-camera Human Tracking via Clustering

Executive Summary

We describe a method for tracking humans in a multi-camera network. Based on online clustering, the proposed method groups single-camera tracks into mutually exclusive subsets corresponding to individuals. By considering all pairwise similarities between single-camera tracks, the method aims to maximise within-cluster similarity, while minimising between-cluster similarity. The method can work in batch mode as well as online mode where it can reassign elements as new data arrives.

Using a synthetic map and agents, and assumed distributions of similarity scores, the behaviour of the algorithm is examined under various conditions such as: number of targets, number of cameras, percentage of camera overlap, probability of track breakage. In order to assess the quality of the tracking, we use a novel error measure based on the mutual information between the ground truth and the computed tracking result.

Experiments show that given a reasonable quality of single-camera tracks, the algorithm is capable of achieving strong tracking accuracy on a number of simulated scenarios. In particular, it was found that tracking accuracy improves as: the number of non-overlapping cameras and targets decreases, more tracks are received, and cameras have higher percentage of overlap. These findings indicate that the proposed method may be suitable for real-world surveillance systems that are of interest. The development of such systems is the focus of our future work.

THIS PAGE IS INTENTIONALLY BLANK

Author

Dmitri Kamenetsky
Intelligence, Surveillance and Reconnaissance Division

Dmitri Kamenetsky obtained a Bachelor of Science with first class Honours in Computer Science from University of Tasmania in 2005. He then completed a PhD in statistical machine learning at the Australian National University and NICTA in 2009. His PhD thesis investigated methods for inference and parameter estimation in graphical models, in particular the Ising model. In 2010, he joined the Intelligence, Surveillance and Reconnaissance Division of the Defence Science and Technology Organisation. His research focuses on multi-camera tracking and cross-matching in video.

THIS PAGE IS INTENTIONALLY BLANK

Contents

THIS PAGE IS INTENTIONALLY BLANK

1 Introduction

The task at hand is to track humans across multiple cameras. We assume the most general case, where no knowledge of the camera network topology exists, the cameras are not calibrated and may overlap or may not overlap in their fields of view. Note that it is generally assumed and required in the literature that camera views either overlap or do not overlap, there is very little existing work on the heterogeneous case. Each camera detects and tracks human targets producing a set of single-camera tracks. These single-camera tracks must be associated with each other to form multi-camera tracks that cover all cameras.

The algorithm must be tolerant of single-camera tracking errors since this is by no means a solved problem. Also the algorithm must be an online method that can be updated incrementally as data arrives in real time. Most of the methods described in the literature are batch methods meaning that all data from past, present and future are processed at once to give a globally optimal solution. This is not practical for systems where the user wants to know the whereabouts of an individual in the camera network in real time.

Here we propose an online clustering method that groups single-camera tracks into mutually exclusive multi-camera tracks, *i.e.,* one for each individual. By considering all pairwise similarities between single-camera tracks, the method aims to maximise within-cluster similarity, while minimising between-cluster similarity.

2 Problem Definition

Let $T = \{t_1, \ldots, t_N\}$ be an ordered set of N observed *single-camera tracks* from all cameras. A single-camera track (or simply *track*) t is a description of the path that a single *target* (human) $\text{tar}(t)$ takes in a single camera $\text{cam}(t)$ field of view. We assume that $\text{cam}(t)$ is known, while $\text{tar}(t)$ is not. Each track t contains information such as:

- Track start time $\text{start}(t)$.
- Track end time $\text{end}(t) \geq \text{start}(t)$.
- Set of target detections $\det(t)$. Each detection $d \in \det(t)$ is also associated with a target $\text{tar}(d)$ and camera $\text{cam}(d)$ (equal to $\text{cam}(t)$).

Our task is to partition T into M mutually exclusive subsets $C = \{C_1, \ldots, C_M\}$ such that

$$\bigcup\nolimits_i C_i = T \quad \text{and} \quad C_i \cap C_j = \emptyset \quad \forall i \neq j.$$

Cluster $C_i \in C$ contains all the tracks corresponding to a single target i and together they form a *multi-camera track*. Hence C_i can be used to track target i in all the cameras. Note that each track $t \in T$ belongs to exactly one cluster in C, which we will call $C(t)$.

3 Clustering

We now describe a clustering method that aims to find the partitioning C. The method is online meaning that it can work as new tracks arrive. Given T, our task is to find the optimal clustering C^*:

$$C^* = \underset{C}{\operatorname{argmax}}\, S(C), \tag{1}$$

where $S(C)$ is some scoring function for clustering C. What form should S have? We know that tracks that are in the same cluster should correspond to the same target and hence must be similar. Tracks that are in different clusters, on the other hand, must be dissimilar. Suppose we have the similarity function for track pairs $S^t : T \times T \rightarrow [0, 1]$, such that $S^t(t_i, t_k)$ is 1 when t_i and t_k are most similar. The score of a clustering C can now be written as:

$$S(C) = \sum_{C(t_i)=C(t_k)} S^t(t_i, t_k) \;+ \sum_{C(t_i)\neq C(t_k)} (1 - S^t(t_i, t_k)) \quad \forall t_i \neq t_k \in T. \tag{2}$$

Note that $S(C)$ is defined to be 0 if $|C| <= 1$. $S(C)$ requires $O(N^2)$ computations, which can be costly when the number of tracks N is large. We now describe an efficient method of computing $S(C)$ incrementally as C is slowly altered.

Algorithm 1 assigns a newly arrived track t to a new cluster in C. It then allows each track to move between clusters. In particular, we consider moving each track p from its current cluster $C(p)$ to some new cluster C_i. If a move increases the score $S(C)$ then we keep it, otherwise we reject it. We terminate when no single move increases the score. Other types of moves can also be considered, however, experiments have shown that this move is sufficient for finding a high-scoring clustering.

Algorithm 1 : Clustering algorithm

Input: Current clustering C, newly arrived track t

1. $C^* := \emptyset$
2. Do :
3. improved := False
4. $\forall p \in T$:
5. $\forall C_i \in (C \cup \{t\})$:
6. Move p to C_i : $C(p) := C(p) \backslash p,\ C_i := C_i \cup p$
7. If $S(C) > S(C^*)$:
8. $C^* := C$
9. improved := True
10. Undo move: $C(p) := C(p) \cup p,\ C_i := C_i \backslash p$
11. While improved

Output: Optimal clustering C^*

It is important to note that we can avoid computing $S(C)$ from scratch in step 7 of Algorithm 1. This is because when we move p from $C(p)$ to some C_i only a small number of terms in (2) change. For example, consider Figure 1. As p is moved, only the contribution from the blue and

red edges changes in the computation of $S(C)$; the contribution from all other edges (dashed) is unchanged. In particular, the contribution from inner-cluster edges (blue) changes from $+S^t(p,\cdot)$ to $1 - S^t(p,\cdot)$, while the contribution from inter-cluster edges (red) changes from $1 - S^t(p,\cdot)$ to $+S^t(p,\cdot)$.

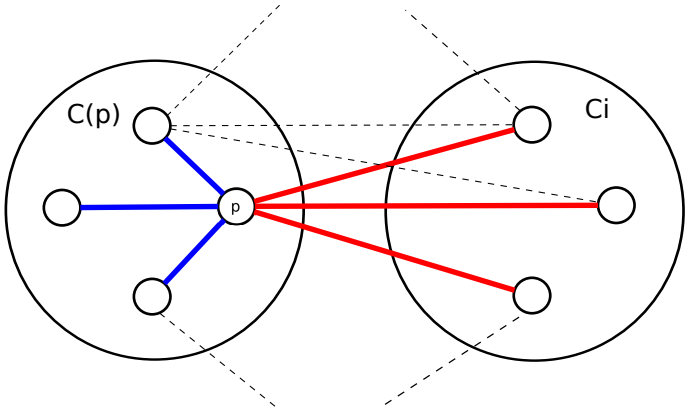

Figure 1: *Track p is moved from its cluster $C(p)$ to a new cluster C_i. Edges between p and tracks in the same cluster are in blue, while edges between p and tracks in C_i are in red. Dashed edges are all the remaining edges (only some are shown for clarity).*

We will now provide an empirical estimate for the running time complexity of Algorithm 1. During each iteration (steps 4 and 5) we try to move every track to every cluster, hence there are NM such moves. In the worst case each cluster contains N tracks, thus for each such move we need to update the contribution of $2N-1$ edges. Empirically the total number of iterations (step 2) seems to grow as $O(M)$. This gives an overall running time complexity of $O(M^2N^2)$. Note that the running time can be reduced significantly by limiting the set of tracks that can be moved during each iteration. For example, we can avoid moving old tracks.

3.1 Related Work

In traditional online multi-camera tracking there is a similarity score $S'(t, C_i)$, which measures the similarity between the newly added track t and an existing multi-camera track C_i. Usually $S'(t, C_i)$ is defined as $S^t(t, p)$, where p is the most recent track in C_i. The new track t is then added to the most similar multi-camera track, as shown in Algorithm 2. Once t has been added to a multi-camera track it remains there forever. This means that "mistakes" made early in the process (such as assigning t to the wrong C_i) cannot be undone later and undermine the quality of future assignments.

In contrast to the traditional method, our method (Algorithm 1) optimises the (dis)similarity between clusters in C. Furthermore, our method is able to modify past "mistakes", allowing it to have the most accurate assignment at any given moment in time.

Algorithm 2 : Traditional online multi-camera tracking algorithm

Input: Current multi-camera tracks C, newly arrived track t

1. bestScore := $-\infty$
2. $b := 0$
3. $\forall C_i \in (C \cup \{\})$:
4. If $S'(t, C_i) >$ bestScore Then
5. bestScore := $S'(t, C_i)$
6. $b := i$
7. $C_b := C_b \cup t$

Output: Optimal multi-camera tracks C

4 Experiments

We used synthetic data to test the clustering algorithms. We simulated agents moving in a 2D world that is observed by cameras with rectangular fields of view (Figure 2). The camera views may or may not overlap. During each time step, each agent produces a new detection by moving to a neighbouring unoccupied location chosen at random. An agent may exit the map if it is located near the map's edge (*e.g.,* pink agent in Figure 2). Each time step there is a probability of 50% that a new agent will appear somewhere on the map.[1] A new track is started whenever an agent enters a new camera. An active track is terminated whenever an agent exits the camera of interest (or exits the map). We use a small probability $\sigma = 10\%$ that an active track will be terminated due to malfunctions in the single-camera tracking algorithm. When this happens, a new track is started where the previous one got broken. Old tracks have little correspondence with recent tracks, even if they are from the same target. For example, imagine trying to match a 2-year old track with a 1-hour old track of the same person. For this purpose, our simulation removes all tracks that are older than W events. Removed tracks do not affect the clustering score nor the performance error (see Equation 5). In our simulations we set W to 25.

Since a track can grow arbitrarily large, it is not possible to store every detection. Furthermore, old detections in a long track have little (if any) correspondence with recent detections. To reflect this we model a track as a first-in-first-out (FIFO) list of detections of maximum size Z. This means that we only store Z of the most recent detections per track. In our simulations we set Z to 25. We compute the similarity of a pair of tracks $S^t(t_i, t_k)$ based on the similarity of their corresponding detections:

$$S^t(t_i, t_k) := \frac{1}{|\text{det}(t_i)||\text{det}(t_k)|} \sum_{d_i \in \text{det}(t_i),\ d_k \in \text{det}(t_k)} S^d(d_i, d_k), \tag{3}$$

where $S^d(d_i, d_k)$ is the similarity between detections d_i and d_k in the range $[0, 1]$. In the design of S^d we took into account three factors: whether d_i and d_k belong to the same target, whether they occur in the same camera, and also their time difference δ. Intuitively, $S^d(d_i, d_k)$ should be higher when $\text{tar}(d_i) = \text{tar}(d_k)$ and lower otherwise. In a real-world application we expect the accuracy of $S^d(d_i, d_k)$ to be greater when $\text{cam}(d_i) = \text{cam}(d_k)$ and lower otherwise. Also the accuracy of $S^d(d_i, d_k)$ should decrease as the time difference δ between the two events increases, since the two

[1] This is provided that the maximum number of agents has not been reached already.

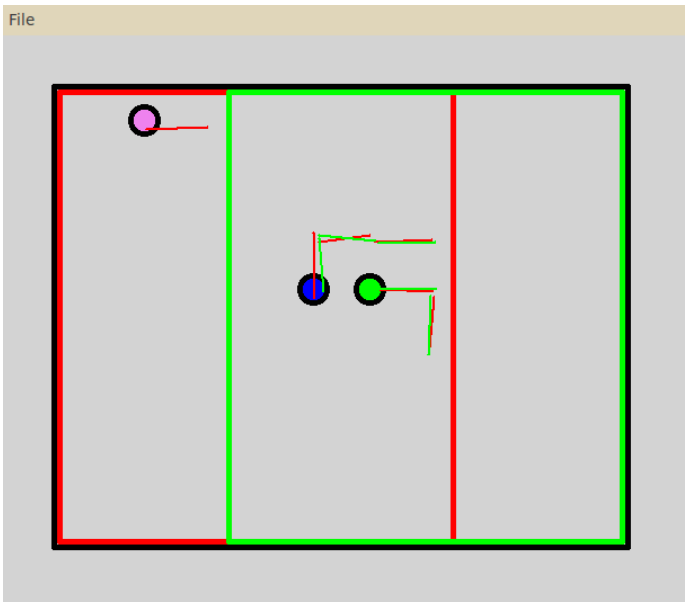

Figure 2: *Simulator example. This 2D world contains 3 agents (pink, green and blue) and 2 overlapping cameras (red and green rectangles). Active tracks are shown as lines coloured as their corresponding camera. For example, green and blue agents are currently being tracked in both cameras, while the pink agent is only tracked in a single camera.*

cameras $\text{cam}(d_i)$ and $\text{cam}(d_k)$ are more likely to be experiencing different lighting conditions.[2] With this in mind, we use four similarity distributions as shown in Figure 3(a). These distributions were chosen because they can be easily sampled. If u is uniformly chosen from $[0, 1]$ then the similarity measure is given as:

$$S^d(d_i, d_k) = \begin{cases} u^{1/(3-0.25\log(\delta+1))} & \text{if } \text{tar}(d_i) = \text{tar}(d_k) \text{ and } \text{cam}(d_i) = \text{cam}(d_k) \\ u^{1/(2-0.25\log(\delta+1))} & \text{if } \text{tar}(d_i) = \text{tar}(d_k) \text{ and } \text{cam}(d_i) \neq \text{cam}(d_k) \\ 1 - u^{1/(3-0.25\log(\delta+1))} & \text{if } \text{tar}(d_i) \neq \text{tar}(d_k) \text{ and } \text{cam}(d_i) = \text{cam}(d_k) \\ 1 - u^{1/(2-0.25\log(\delta+1))} & \text{if } \text{tar}(d_i) \neq \text{tar}(d_k) \text{ and } \text{cam}(d_i) \neq \text{cam}(d_k) \end{cases} \quad (4)$$

In the above, δ affects the width of the distributions as shown in Figure 3(b).

In the experiments we sample $S^d(\cdot, \cdot)$ from one of the four relevant distributions. Once a value has been chosen it is fixed and does not change over time. In contrast, the value of $S^t(t_i, t_k)$ may change over time as new detections are added to t_i or t_k. As a result, the value of $S^t(t_i, t_k)$ only converges to its true value over time.

Note if t_i and t_k overlap in time[3] and they are from the same camera, then we know that they cannot belong to the same multi-camera track and hence we set $S^t(t_i, t_k) := -\infty$. Every experiment is evaluated on 100 trials. If $\hat{C}$ is the true clustering for the set of tracks T, then the error for the computed clustering C is:

[2]For example, one camera is filming during the day, while the other camera is filming during the evening.
[3]This occurs if $\text{start}(t_i) \leq \text{end}(t_k)$ and $\text{start}(t_k) \leq \text{end}(t_i)$.

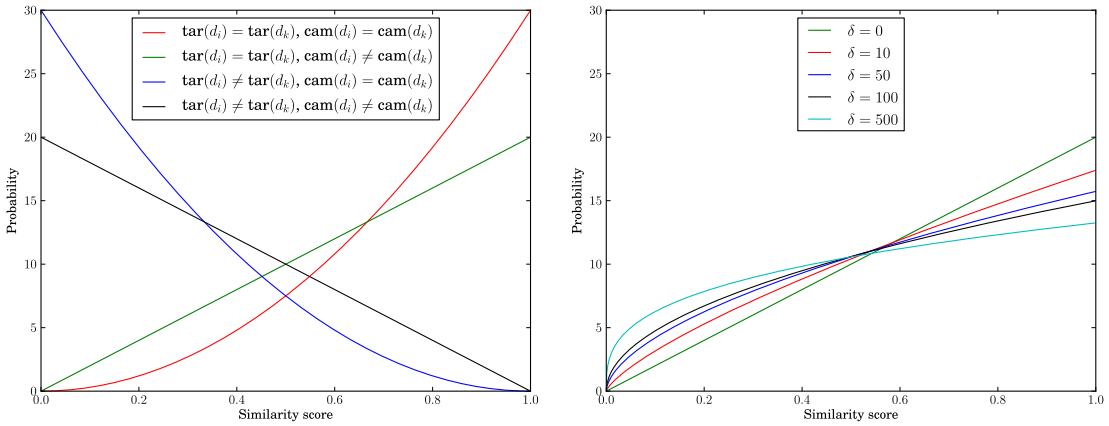

Figure 3: *(a) The four distributions that we sampled to obtain the similarity score $S^d(d_i, d_k)$, assuming the two detections occurred at the same time,* i.e., *$\delta = 0$. (b) The effect on one of the distributions ($tar(d_i) = tar(d_k)$ and $cam(d_i) \neq cam(d_k)$) as the time difference δ is varied from 0 to 500.*

$$E(C, \hat{C}) := \frac{2H(C, \hat{C}) - H(C) - H(\hat{C})}{H(C, \hat{C})}, \text{ where} \tag{5}$$

$$H(C) := -\sum_{C_i \in C} \frac{|C_i|}{N} \log_2 \frac{|C_i|}{N} \text{ and}$$

$$H(C, \hat{C}) := -\sum_{C_i \in C,\ \hat{C}_k \in \hat{C}} \frac{|C_i \cap \hat{C}_k|}{N} \log_2 \frac{|C_i \cap \hat{C}_k|}{N}.$$

$H(C)$ is the entropy of C, while $H(C, \hat{C})$ is the joint entropy of C and $\hat{C}$. The numerator of $E(C, \hat{C})$ is known as *variation of information* or *shared information distance*. E is a *metric*[4], with $E(X, X) = 0$ and $E(X, Y) \leq 1 \ \ \forall X, Y$.

We investigated how various factors affect the performance of the clustering method in Algorithm 1:

- Varying the number of agents and 4 cameras: Figures 4(a-b).
- Varying the number of non-overlapping cameras and 5 agents: Figures 4(c-d).
- Varying the percentage of camera overlap for 2 cameras and 5 agents: Figures 4(e-f).
- Varying the probability of track breakage σ for 2 cameras and 5 agents: Figures 5(a-b).
- Varying the maximum track size Z for 4 cameras and 5 agents: Figures 5(c-d).
- Varying the maximum age of a track W for 4 cameras and 5 agents: Figures 5(e-f).

[4] A metric is a binary function that satisfies symmetry, positivity and the triangle inequality.

For all experiments $\sigma = 10\%$, $W = 25$ and $Z = 25$ unless stated otherwise. In general the errors decrease over time. This can be explained by the fact that clusters become larger, the (dis)similarity between clusters is estimated more accurately and the effect of noise in the similarity scores is reduced. Under various conditions the clustering method is able to recover and maintain good clustering. The clustering performance improves as the percentage of camera overlap increases (Figures 4(e-f)). Higher percentage of camera overlap results in a higher rate of detections and hence provides more information to the clustering algorithm. Results in Figure 4 are as expected: the error increases as the number of agents increases, the number of non-overlapping cameras increases, or the percentage of camera overlap decreases. It may seem that a larger number of non-overlapping cameras should decrease the error since each target is seen more times. However this is not the case, because a larger number of non-overlapping cameras will produce a larger number of tracks N - increasing the complexity of the problem we are trying to solve. Figures 5(a-b) show that an increased percentage of track breakage does not affect the tracking accuracy - this seems counterintuitive and at this stage we cannot explain it. Figures 5(c-d) show that it is sufficient to have a rather small maximum track size of 5 detections. Figures 5(e-f) show that the error decreases significantly as W increases. From our experiments, the use of the constraint $S^t(t_i, t_k) = -\infty$ for simultaneous tracks gives only a marginal improvement in error.

5 Future work

If we know that there are exactly M targets then the clustering task can be formulated via an M-labeled graphical model, which can be solved efficiently. Let $G(T, \mathcal{E})$ be a fully connected graph of tracks (*i.e.,* $\mathcal{E} = T \times T$) and each track t_i takes the label $y_i \in \{1, \ldots, M\}$, then we minimise the following real-valued energy function:

$$E(\mathbf{y}) := \sum_{(i,j)\in\mathcal{E}} [\![y_i \neq y_j]\!]\, S^t(t_i, t_j), \tag{6}$$

where $[\![x]\!]$ is an indicator function that is 0 if x = True, and 1 otherwise. Minimising (6) is now equivalent to maximising (2). Importantly, the energy function (6) can be minimised using multiway cuts. This will give us $\mathbf{y}^* = \text{argmin}_{\mathbf{y}}\, E(\mathbf{y})$ and from this we can obtain the clustering C^* via a simple Depth First Search. Note that for $M = 2$ this approach becomes the standard graph cuts algorithm, which is exact and has $O(N^3)$ running time complexity [Kolmogorov & Zabih 2004]. Our future work will focus on using the proposed method with real-world surveillance systems that are of interest

6 Conclusion

We presented a method for tracking humans in a multi-camera network. Our method is based on online clustering, where single-camera tracks are grouped into mutually exclusive subsets corresponding to individuals. Clustering is achieved by maximising a cost function via a series of swap moves. Importantly, the cost function takes into account all the pairwise similarities between single-camera tracks, thus utilising all the available information. The method can work in batch mode, as well as, online mode where it can reassign elements as new data arrives.

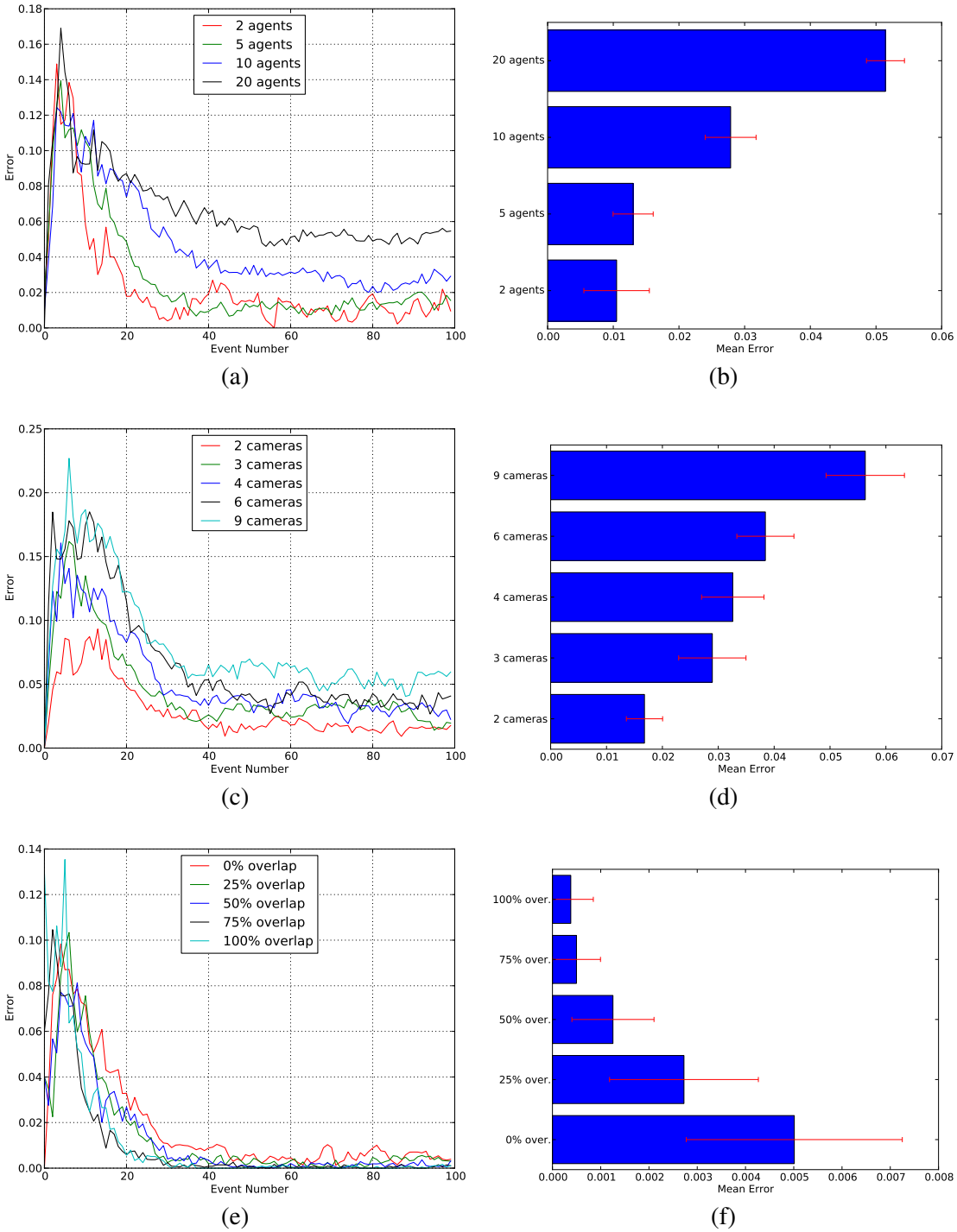

Figure 4: *Error obtained by the clustering method for various parameter settings. Left figures show the change of error through time. Right figures show the mean and variance of error after 50 events. (a-b) Varying number of agents and 4 cameras. (c-d) Varying number of non-overlapping cameras and 5 agents. (e-f) Varying percentage of camera overlap for 2 cameras and 5 agents.*

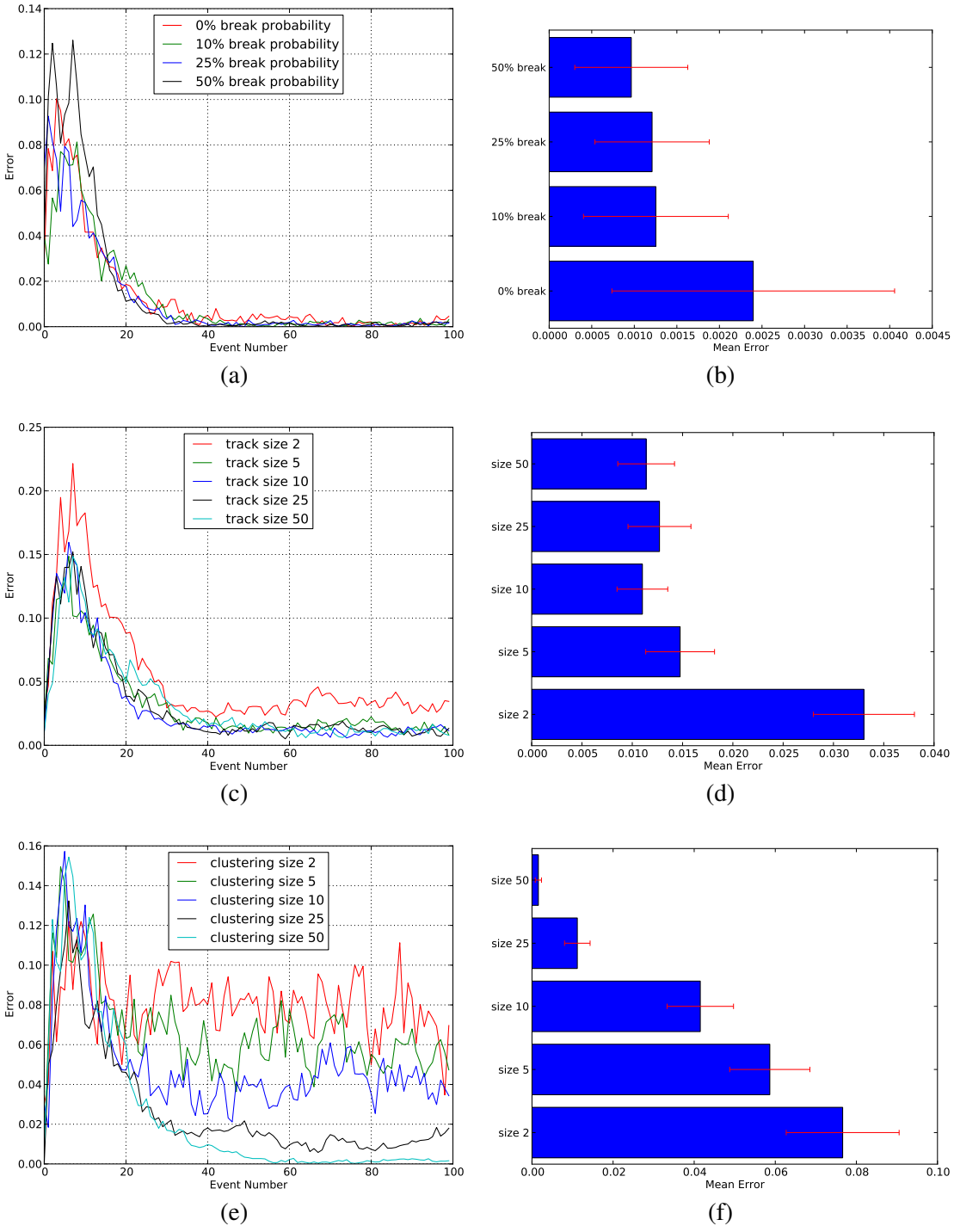

Figure 5: *Error obtained by the clustering method for various parameter settings. Left figures show the change of error through time. Right figures show the mean and variance of error after 50 events. (a-b) Varying the probability of track breakage σ for 2 cameras and 5 agents. (c-d) Varying the maximum track size Z for 4 cameras and 5 agents. (e-f) Varying the maximum age of a track W for 4 cameras and 5 agents.*

We used synthetic camera topology maps to examine the behaviour of the algorithm under various simulated conditions. By varying each input parameter we were able to characterise the performace of the algorithm. Given a reasonable quality of single-camera tracks, the algorithm was able to achieve strong tracking accuracy on a number of simulated scenarios. In particular, it was found that tracking accuracy improves as: the number of non-overlapping cameras and targets decreases, more tracks are received, and cameras have higher percentage of overlap.

References

Kolmogorov, V. & Zabih, R. (2004) What energy functions can be minimized via graph cuts?, *IEEE Transactions on Pattern Analysis and Machine Intelligence* **26**, 65–81.

DEFENCE SCIENCE AND TECHNOLOGY ORGANISATION DOCUMENT CONTROL DATA				1. CAVEAT/PRIVACY MARKING
2. TITLE Multi-camera Human Tracking via Clustering			3. SECURITY CLASSIFICATION Document (U) Title (U) Abstract (U)	
4. AUTHOR Dmitri Kamenetsky			5. CORPORATE AUTHOR Defence Science and Technology Organisation PO Box 1500 Edinburgh, South Australia 5111, Australia	
6a. DSTO NUMBER DSTO–TR–2663	6b. AR NUMBER AR 015-218		6c. TYPE OF REPORT Technical Report	7. DOCUMENT DATE November 2011
8. FILE NUMBER 2011/1159124/1	9. TASK NUMBER NS 07/114	10. TASK SPONSOR DSTO	11. No. OF PAGES 10	12. No. OF REFS 1
13. URL OF ELECTRONIC VERSION `http://www.dsto.defence.gov.au/publications/scientific.php`			14. RELEASE AUTHORITY Chief, Intelligence, Surveillance and Reconnaissance Division	
15. SECONDARY RELEASE STATEMENT OF THIS DOCUMENT *Approved for Public Release* OVERSEAS ENQUIRIES OUTSIDE STATED LIMITATIONS SHOULD BE REFERRED THROUGH DOCUMENT EXCHANGE, PO BOX 1500, EDINBURGH, SOUTH AUSTRALIA 5111				
16. DELIBERATE ANNOUNCEMENT No Limitations				
17. CITATION IN OTHER DOCUMENTS No Limitations				
18. DSTO RESEARCH LIBRARY THESAURUS Cameras, Surveillance, Defence				
19. ABSTRACT We describe an efficient method for tracking humans in a multi-camera network. Based on online clustering, the proposed method groups single-camera tracks into mutually exclusive subsets corresponding to individuals. Experiments indicate that the algorithm is capable of achieving strong tracking accuracy on simulated data.				